E82-10372

AgRISTARS

SR-X2-04260
NAS9-16446

NASA-CR-167618

A Joint Program for Agriculture and Resources Inventory Surveys Through Aerospace Remote Sensing

Supporting Research

March 1982

ALGORITHMS FOR SCENE MODELLING

M. E. Rassbach

(E82-10372) AGRISTARS. SUPPORTING RESEARCH: ALGORITHMS FOR SCENE MODELLING (Elogic, Inc.) 55 p HC A04/MF A01 CSCL 02C N82-32794

Unclas
G3/43 00372

Elogic, Inc.
4242 S.W. Freeway, Suite 304
Houston, Texas 77027

Lyndon B. Johnson Space Center
Houston, Texas 77058

1. Report No. SR-X2-04260	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Supporting Research Project Algorithms for Scene Modelling		5. Report Date March 30, 1982
		6. Performing Organization Code
7. Author(s) M. E. Rassbach		8. Performing Organization Report No. NAS-811
		10. Work Unit No.
9. Performing Organization Name and Address Elogic, Inc. 4242 S.W. Freeway, Suite 304 Houston, Texas 77027		11. Contract or Grant No. NAS9-16446
		13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Lyndon B. Johnson Space Center Houston, Texas 77058 (Tech. Monitor: F. G. Hall)		14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

A detailed analysis is made of the requirements for comprehensive analysis of Landsat scene or other visual data. This leads to a general model of a scene and to a computer algorithm for finding the particular model for a given scene. These are detailed in this report.

17. Key Words (Suggested by Author(s)) Scene analysis Computer vision Robotics Image analysis	18. Distribution Statement

19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 50	22. Price*

*For sale by the National Technical Information Service, Springfield, Virginia 22161

SR-X2-04260
NAS9-16446

ALGORITHMS FOR SCENE MODELLING

BY

M. E. RASSBACH

This report describes Classification activities of the Supporting Research project of the AgRISTARS program.

Elogic, Inc.
4242 S.W. Freeway, Suite 304
Houston, Texas 77027

March 30, 1982

TABLE OF CONTENTS

Algorithms For Scene Modelling

Introduction

In this report we study the concept of a scene model for image analysis. This concept was originally invented for use on Landsat data, but after the model was defined and a program to generate it was designed, it became apparent that the method would apply to robotics, target acquisition, and similar recognition problems. Thus this report has been couched in general terms.

For a general overview of the literature in this area, the reader is referred to the texts in the bibliography.

Overview: The Need for a Scene Model

Various methods have been used or proposed for analyzing visual scenes. Most of these methods are local, that is, they look for local features such as edges, corners, textures, or combinations of these. We may instead build a structure from edges, corners and other features which contains the essential information of the figure in a more explicit form. That is, in such a model, all the edges have already been detected and included, so it is possible to refer to the model rather than reprocessing the original data. Such a model is more likely to be complete, and relations between distant parts of the figure will already have been taken into account. For example, if we are looking for a line within an image, the knowledge that there is exactly one such line will help us tell which of the detected line segments might be the real line.

Building a model of the scene, then, provides necessary infrastructure which allows later recognition algorithms to process the scene easily. In the model, all preliminary calculations have already been done, and the secondary algorithms need only consult a table. In addition, when we construct the model we may impose conditions of consistency which allow us to cull the model and eliminate spurious effects. When an attempt is made to apply the high-level recognition algorithms without this culling, the extra, noise data will also be used in the calculation, resulting in greatly reduced reliability.

Thus the creation of a model decreases noise, increases reliability, cuts computer time, and simplifies the programming of higher-level recognition elements. A scene model appears to be a basic element on which scene analysis may be made.

Parts of the Scene Model

What sort of scene model are we interested in? Clearly the set of models allowed, or the language in which they are expressed, is infinite. We give here a basic model which may be ramified to allow more detail. However, even if all the ramifications are included, there will exist types of scenes for which the model is not applicable, and separate classes of models would have to be made for these. Most practical situations, from Landsat data analysis to robot vision or target identifications, should be amenable to the methods used here. In an intuitive sense, they are based on the characteristics of the human visual system, and thus can handle any problem which can be attacked by that system.

An actual visual scene contains a number of elements. The most prominent of these are boundaries, or sharp demarcations between regions with dissimilar properties, and continuous variations. Boundaries may be very sharp (at the pixel or optical resolution of the system) or they may be diffuse, such as the edge of a shadow. There is usually some degree of similarity between the regions on the two sides of an edge. This is most noticeable for the two sides of a shadow, where the only difference is in luminance. Other edges may change only part of the characteristics of a region. For example, in Landsat data the underlying soil types may give a set of edges which is independent of the edges which delineate crop boundaries.

Similarly for an image printed on paper, the texture and other conditions of the paper are continuous across printed boundaries. There may also be boundaries within an otherwise continuous region, for example, uneven paint, grease spots, or uneven crop growth in a Landsat scene. Usually if there are two sets of variations, the boundaries are independent and cross each other freely.

Continuous variations range from diffuse boundaries to a gradient extending across the whole scene. Perhaps the most common type of gradient is a mottling of an area, where the variation has some characteristic lower spatial frequency. Continuous variations may also occur in parameters besides the brightness, such as parameters involved in texture, or in the statistics of region sizes and shapes. Most of the programs for finding various types of continuous variation can be applied to these other types of data as well. In fact, the boundary recognition algorithms may also be applied to these data sets.

Besides the basic categories of boundaries and continuous variation, there are a number of additional aspects to a reasonable model of a visual scene. The most important of these is texture. This may range from the fine-grained texture of a fabric to the coarse texture of a tree bark. In agricultural remote sensing, texture is not important, since the regions of interest are fairly uniform; however, in remote sensing of wild lands texture may still be meaningful. Texture may also be useful in recognition of urban regions.

One major difficulty with texture is the ambiguity in what is texture and what is structure. An urban region, for example, may be thought of as many small regions or as a large region with urban "texture." Similarly, a robot handling machine parts may interpret a grease spot or a ragged, burred edge as meaningful data or irrelevant information (noise). Most such separations can be made on the basis of the size, distinctness, regularity, etc. However, the criteria to be used will vary from application to application, and the separation of texture from feature may not be possible without more detailed information about the expected visual scene.

Another element of the scene model are geometric relations. The simplest of these are the identification of true circular arcs. Other geometric elements include the parallelism or perpendicularity of lines. This should include mostly parallel curving lines, such as river beds. All of these elements are important, but can probably be relegated to higher levels of analysis.

Describing the Scene Model

The scene modelling program must generate a numerical description of the scene. This must contain a representation of all scene elements, such as boundaries, continuous variations, textural information, etc. The method of description attempted here hierarchical--that is, the boundaries are taken to be basic to any description, and the other parts of the description are hung onto this component. Some continuous variation across boundaries is described independent of the boundary system. Questions such as independent systems of boundaries (e.g. shadows vs. edges) are not attacked at this level, although the approach for doing so is clear.

We wish the numerical model to be the best possible description of the picture, without having so much detail that the main features are difficult to extract. We may also wish it to be in accord with the hierarchical structure of the description, so that the code for the extraction of various features can be written independently. This would allow the edge extraction process to be programmed without the necessity of coding to process texture, continuous variations, etc. Then the higher level analyses could be coded using a well-debugged background routine.

The boundary model consists of a graph of points, edges, and fields, as shown in Fig. 1. Boundaries lie between fields and are conceptually the basic unit. In the simplest form of this model boundaries are restricted

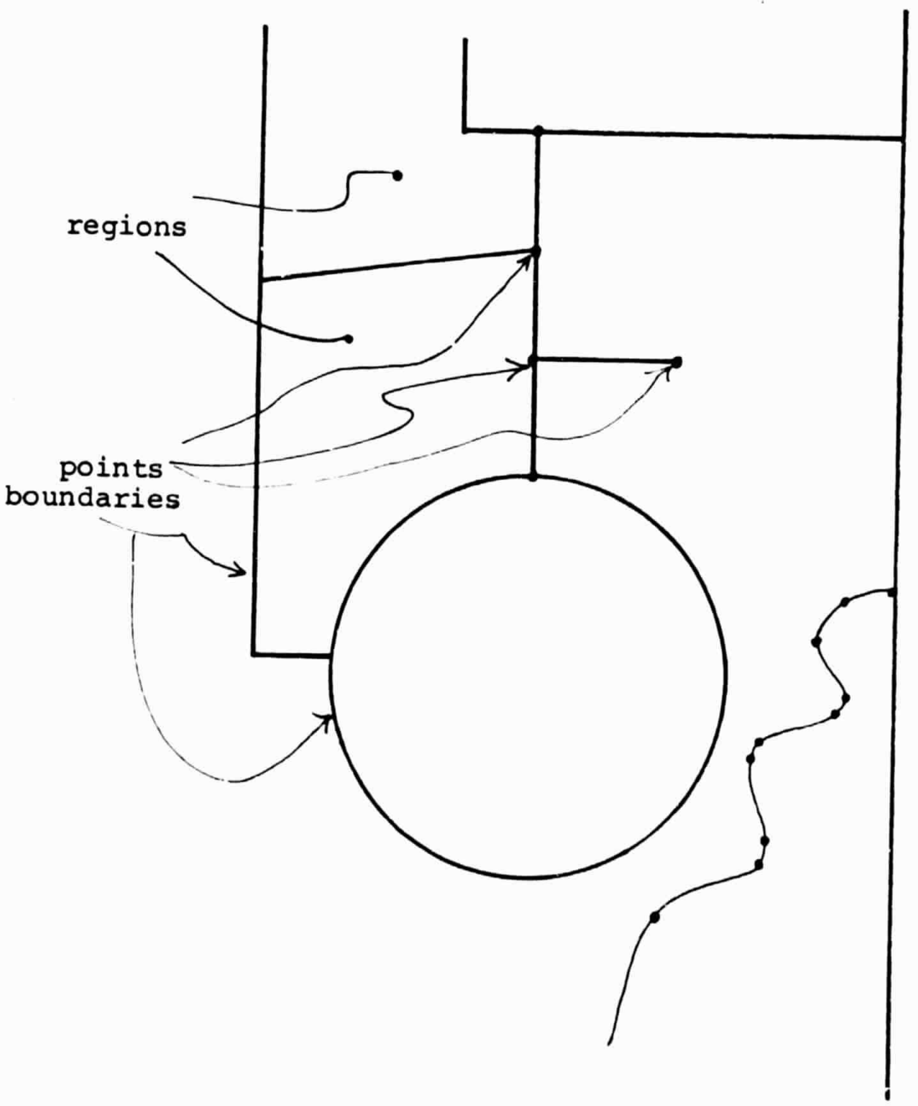


Fig. 1 The basic elements of the boundary model.

to straight lines. Curved lines are then represented by sequences of straight lines, in a manner reminiscent of numerical integration. A higher level representation of lines includes circular arcs as well as straight lines. Non-circular curves would still be represented by a sequence of arcs. In a completely ideal system even these curves would have a direct representation, such as by spline fits.

Points, in the boundary model, refer to the end points of lines. Of course, several lines may have the same end point. Regions are the areas enclosed by lines.

All of these may be represented in the computer using pointer structures. In Fig. 1, each line would be represented by a block of data giving its end points, curvature, line sharpness, and other information. Each point would correspond to a block of information giving its location and a list of lines incident on the point. The data associated with a region need only contain one line or edge bordering on the region, the side of the line which the region is on, and any ancillary information about the region. This ancillary information may include statistical information on the points within the field. For more complete representations which include the continuous variation analysis, the continuous information is accessible through the data for that particular region.

The continuous variation of variables across regions in Figure 1 is represented by a zonal process, described below. It is also possible to describe continuous variations across larger regions or the whole scene. The interrelationship between these global variations and the single region variations has not yet been worked out; however, it should be possible to work out a data structure which can accomodate both types of variation.

The zonal method separates the region into zones of varying size, with the data constant or linearly varying across a zone. The zones are selected to be of the maximum size consistent with no statistically significant variations across the zone. This method provides a general description of the continuously varying data with maximum accuracy and minimum description size.

The procedure for modelling texture is much less well defined. This is partly because there are some fairly complex textures which we might wish to represent, such as a chain-link, or chicken-wire fence. We will restrict this analysis to "local" textures, where the texture can be represented by few-point statistical functions. The one-point textural function is the distribution (mean, variance, etc.) of the individual data points. The two-point function is the power spectral density. Simple 3 and 4-point functions may also be developed. The exact parameters available from a textural analysis depend on the application; a power spectral density or a distribution function contain too many parameters to be useful statistics, so that some condensed form must be used. The actual condensation used will have to be different for different applications. In any event, the texture modelling system will produce a set of local texture parameters which may be processed by the continuous variation system. These parameters may also be processed by the boundary system to find boundaries which appear primarily in texture. Although these boundaries will not be sharp, they are still detectable, and in fact are detectable to the eye.

In summary, we want a numerical description of the data which embodies the major components: boundaries, continuous variation, and texture. This model should be hierarchical, so that we may represent boundaries even if the code for the texture and continuous variation is not yet written.

This description is compact yet fairly complete: the most essential information is represented, without too much inessential information. In terms of volume of data, the major part not carried over into the abstraction is the pixel-to-pixel variations in brightness, generally treated as noise. Of course the original data will still be available for analyses not possible in the model.

The numerical description includes first a description of the boundaries, by giving the end points of boundary line segments. Then continuous variations are represented by zones and gradients. Texture is represented by average textural measures over a region, using the zonal system.

Detailed Description of Boundary Analysis System

In this section we give a more detailed description of the boundary analysis system, both its function and output. This system takes raw pixel data and identifies and locates all the boundaries in the image. This is done in three parts: initialization, boundary adjustment (tuning), and faint and diffuse boundary location. The initialization step goes over the image and finds all the clearly present lines and boundaries, and gives their location to a pixel or so. This is basically a rapid process based on the concept of global edge and line detection, which will be elucidated below. When it is finished there is available a consistent graph of the boundaries, which can be used in further processing.

Although this initial graph is sufficient for many purposes, it is not the optimal or best fit representation of the data. Two types of improvements are possible, and correcting routines exist to make these improvements in the model of the scene. These correcting routines may be applied iteratively, if necessary, in order to get the desired accuracy.

The first correction is to the exact positioning of the points and boundaries. The estimated error of the initializing routine in the positions of the points is currently estimated at about $\pm$ 1 pixel. This causes proportional errors in line positions, region areas, and other geometry. For many applications of a general visual system this is easily adequate. However, when attempting high-resolution analyses, where a pixel is relatively large, it is necessary to tighten the estimation of the point position as much as possible, to a theoretical limit which depends on noise levels, but is on the order of $\pm$ .1 pixel. The routine to do this correction examines the division of each boundary pixel and makes a best fit of the boundary through these split pixels. The actual adjustment is in the boundary end points, requiring several boundary segments to be consistent.

The second correction is the detection of faint and diffuse lines not discovered by the boundary initialization routine. The initialization routine is designed to quickly pick up the obvious lines and edges, not to detect features close to the edge of statistical obscurity. It is thus necessary to go over the regions produced by the initialization routine and determine if there are any faint edges or boundaries which are so spread out and gradual that they were not detected during the initialization stage. The program to do this uses the continuous variation (zonal) system to determine if statistically significant variations still exist in the region. If there are such variations, then, the faint and diffuse edges program will determine the optimum location of an edge, and whether the edge fits the data sufficiently better than a gradual variation. If so, a new edge has been detected. The detection program will process all regions in this fashion, finding faint and diffuse boundaries, locating their end points, and putting them in the boundary graph. It may be run recursively applied to regions divided during the first pass. Locating faint lines (not edges) is more difficult and will require a more complex routine than given here.

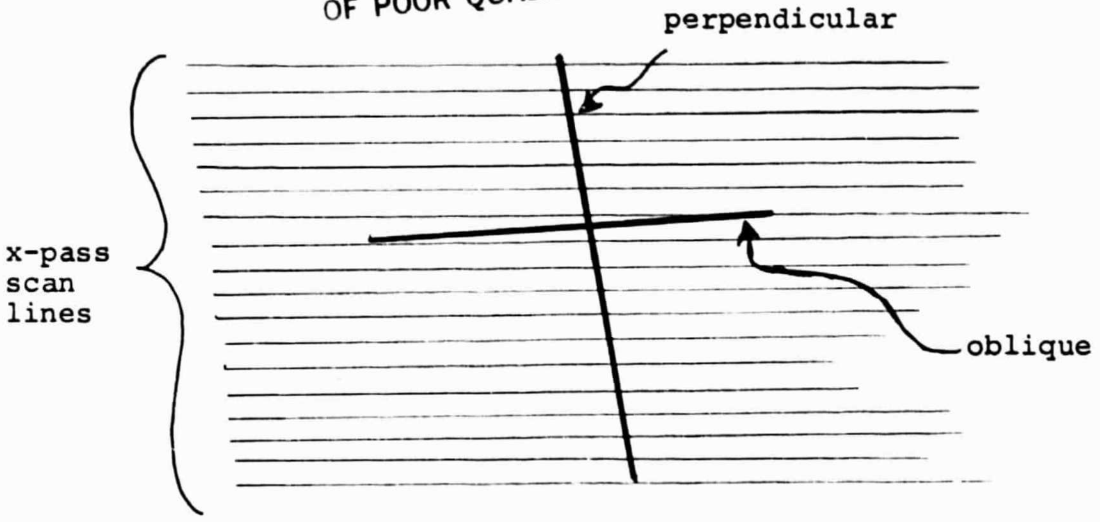
In summary, the boundary analysis program has three parts, initialization, fine adjustment, and faint and diffuse edge detection. Its output is a graph of the boundaries and lines within a figure. The initialization routine alone will produce the same type of output, although the actual values will not be as refined.

Initialization Routine

The boundary initialization routine is a basic sequential procedure on the incoming (multispectral) pixel brightness values. It quickly goes through the image locating all the essential and prominent lines and edges. It will locate nearly all statistically significant sharp lines and edges, and the stronger diffuse lines and edges. Boundary positioning should be more accurate than $\pm$ 1 pixel.

The procedure is a simple and direct algorithm processing the points in a raster scan with a minimum number of operations per point. In the currently planned version, an additional scan is made with the x and y axes interchanged, in order to process oblique lines not easily detected in the first pass. In a later, or production version this second pass could be incorporated in the first, resulting in better data flow. These two passes are referred to as the x-pass and y-pass respectively.

The use of two passes is made necessary by the difficulty of detecting highly oblique lines. Fig. 3 shows the x-pass and y-pass scans including a normal (perpendicular) and oblique line. The oblique line shows only a gradually changing profile in the x-direction scan, so that it will not be readily detected. The y-direction scan will readily detect this line but will not see the vertical line detected by the x-scan. For this reason the lines detected by the x-direction scan are arbitrarily limited to about 45° slope. Since the vertical and horizontal lines are detected in different passes, special processing must be done in a



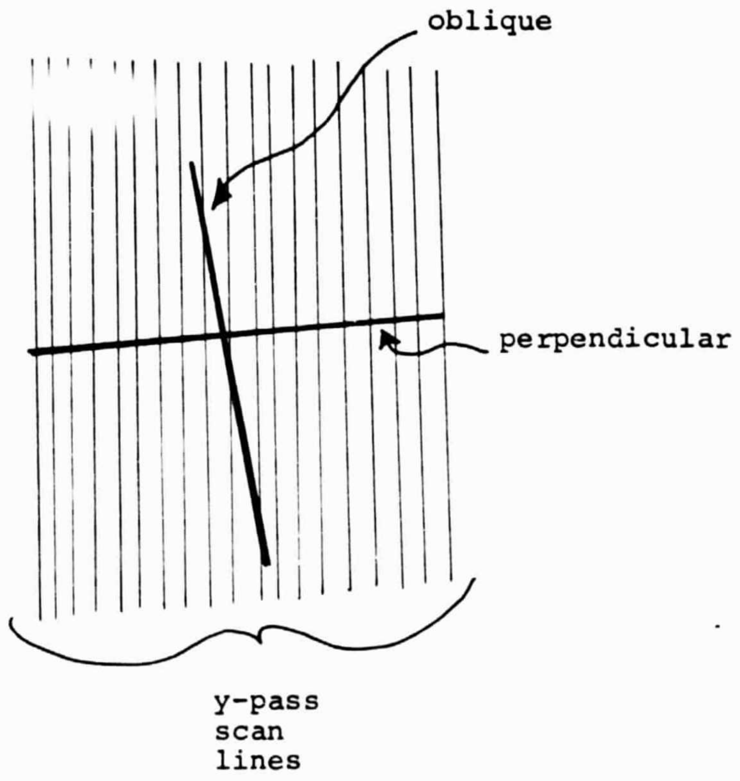


Fig. 3. (Upper) X-pass scan with perpendicular and oblique lines. (Lower) Y-pass scan with perpendicular and oblique lines.

a later stage to associate these two groups of boundaries.

The analysis below applies to the x-pass. The y-pass is the same, mutatis mutandi, but has some extra steps to allow coordination with the previously calculated x-pass results. This will be detailed after the x-pass discussion. The data is processed in a row-by-row fashion. Certain conditions are tested for at each point to determine whether it is a previously undetected edge, or, if a nearby point on the row above was an edge, whether this point is the continuation of the edge, whether the edge is turned into two edges, whether the edge angle is within the $\pm 45^\circ$ limit, etc. These tests are made for each point in the row as the program processes it. When the tests are satisfied, an appropriate action is taken in the model being created. If the model boundaries are depicted by lines, then these are filled in row-by-row as though the model were being painted by a wide brush.

The tests made and the appropriate actions are listed in Table 1a.

This set of tests is used to build up the graphical model of the boundaries from the numerical image. Lines are started, stopped, and joined using these transformations. When the image has been completely scanned, a complete graph of the lines with slopes less than 45° will have been generated.

The x-pass leaves special markers to be picked up by the y-pass. This is to allow the y-pass logic to pick up certain conditions or vertices where one line is

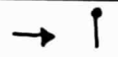
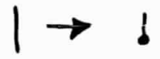
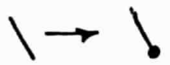
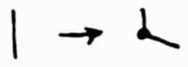
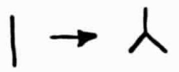
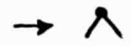
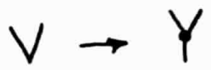
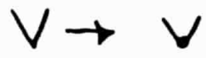
Transform	Criteria	Action
Start a line	Edge or accumulated edge too large	Start line (blank) 
End a line	Edge accumulation goes to zero	End line 
Agle excess	Line angle >> limit (45^{0})	End line 
Bend line	Edge angle (or curvature) exceeds threshold	Finish line, start new line 
Fork	Line spread increases	Divide line into two 
Top Corner	(Fork with short upper segment)	Start two new lines (blank) 
Junction	Two lines intersect	Finish lines, start new line 
Bottom corner	Too short a line below a junction	(Terminate new line) Equivalent to: Finish lines 

Table la. Transformations made during x and y passes.

generated during the x-pass and the other line is generated during the y-pass. These lines must be joined to form the full graph. Table 1b. gives these transformations.

When the scene is processed transforming the graph according to these rules, the result will be a graph of the boundaries in the segment. The details of this graph depend in part upon the type of basic line detector used. The detectors and their thresholds for line detection and continuation are an important part of this graph-building process. This will be taken up in the section Global Edge Detection.

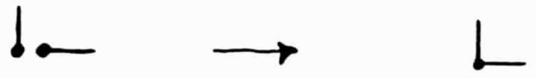
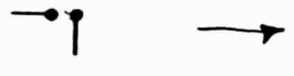
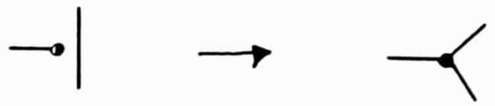
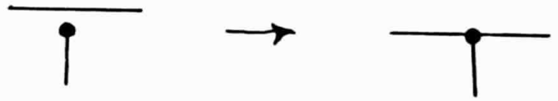
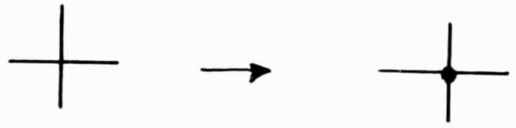
Transform	Criteria	
Join a y-start to x-start or end	proximity	
Join a y-end to x-start or end	proximity	
Join a y-start or end to an x-line	proximity/intersection	
Join an x-start or end to a y-line	proximity	
Join intersecting x and y lines	intersection	

Table 1b. Special transformations used only on y-pass.

ORIGINAL PAGE IS OF POOR QUALITY

Summary of Boundary Initialization Procedure

The boundary initialization procedure is a program for obtaining the basic set of boundaries in an image. It does this using simple point-by-point procedures. Two passes are made, one in the x-direction, the other in the y-direction, although these might be combined in later versions. As the program scans over the image in each pass, transformations are made in sequence on the boundary graph for the image, until the entire graph has been built. This graph, after both passes, is then the output of the boundary initialization routine, which may be passed directly into routines doing classification of areas, shape analysis, etc. Alternatively, this output may be passed to additional routines which improve the quality of the boundary graph before passing the output to the final processing routines.

Global Edge Detection

The boundary initialization routine uses a special form of edge detection called global edge detection. Most edge detectors are local, that is, the condition of the line passing through a pixel is detected by looking at some region around the pixel and doing a fixed calculation. A global edge detector attempts to detect a whole line segment as a single unit. Figures 4a and 4b show a local and a global detector respectively.

A global detector is superior to a local detector in several ways. First, more information is integrated into the final decision as to whether a line is present or not. Second, in detecting an actual line using local detectors, a local detector must be applied to each point on the line. (See Figure 4c) This means that data from each point must be used more than once in doing the detection. Further more, an additional algorithm is required to assemble the edge pixels into meaningful lines. Thus a local detector plus this assembly algorithm becomes a two-stage, poorly optimized, global detector. In addition, a global detector looks only at one row at a time, and so will require less computer time.

A global detector is intrinsically closer to what is meant by an edge detector than is a local detector. An edge is not a point object, but has the geometry of a line, that is it extends over a group of pixels. Since a line cannot be separated from its continuations above and below a given pixel, a detection mechanism which takes the whole

ORIGINAL PAGE IS
OF POOR QUALITY

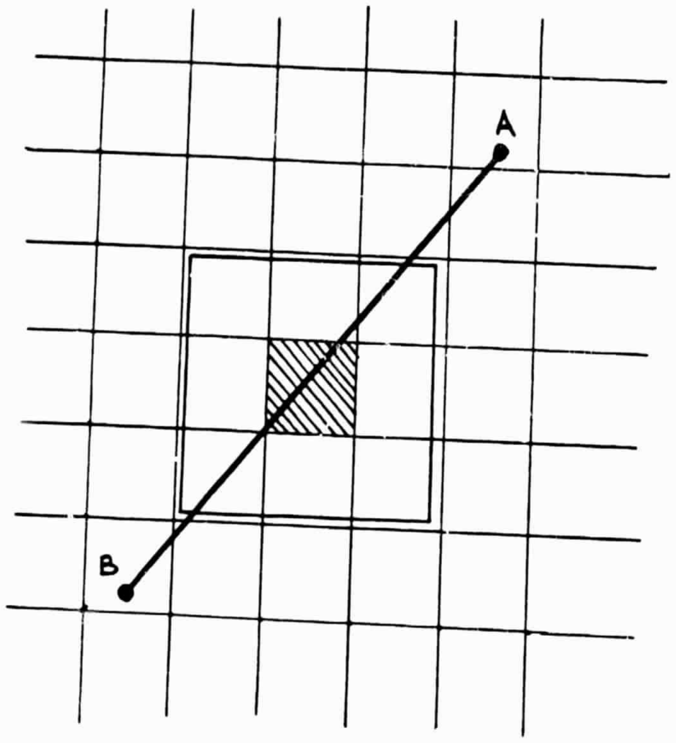


Fig. 4a. Local edge detector. Detects an edge through a pixel (shaded) by looking at a local region around it (3 x 3 square).

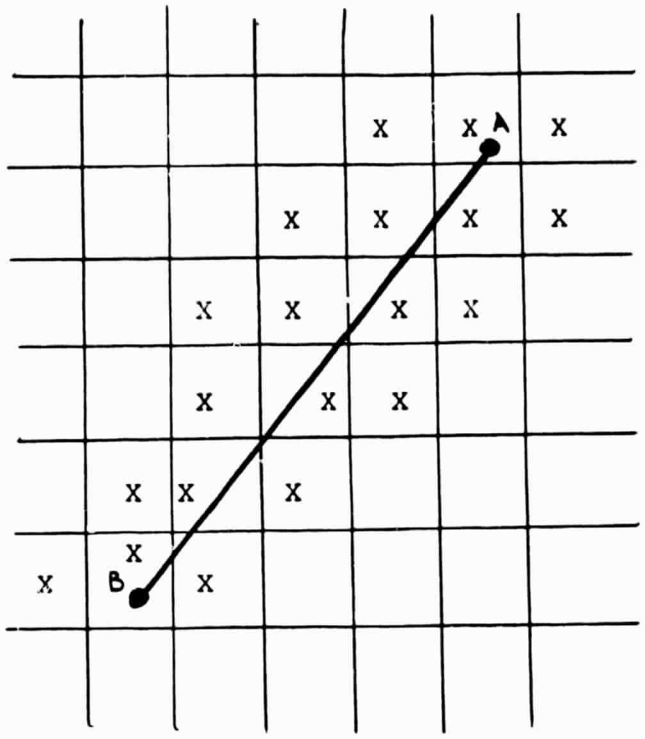


Fig. 4b. Global edge detector. Detects a line from A to B by looking at all the x'd pixels. Information can generally be accumulated along the line, without any unnecessary calculation.

ORIGINAL PAGE IS
OF POOR QUALITY

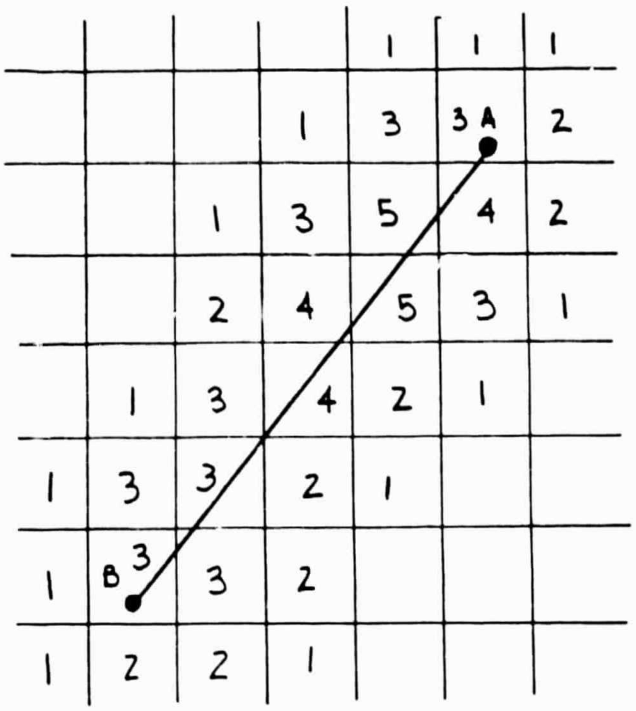


Fig. 4c. Number of times each pixel is used in detecting a line from A to B. The number of times each pixel is used in the process of detecting a line from A to B on a pixel by pixel basis. The single-pixel edge detectors along the line are usually treated as independent, but in fact are not, as shown by this figure, by the high degree of overlap.

line into account is much closer to the actual situation than is an algorithm which looks at each pixel separately. This proximity to the real problem makes the global edge detector approach superior to a set of one-pixel edge detections. Not only is the data processed in a fashion which does not use the data from a single pixel more than once, but also the question of the existence of a line becomes a well-posed statistical problem.

In principle, to detect a line from each point A to each other point B would require $N^2 \ell$ calculations, where N is the number of terminal pixels, and ℓ is the line length. In fact, however, most of these calculations are duplicate, since the sums for one line are parts of the sums for a subline, as shown in **Fig.** 5. The net effect of this is to reduce the order of the calculation to N, the number of pixels in the scene. The global edge detection process is reduced to a simple scan of the image doing the appropriate calculations. These calculations consist of looking for the beginnings of lines, continuing lines, ending lines, and some combinations of these processes. These are the transformations given in Table 1. Accumulations are made of the edge strength of each line, and of its best starting and ending points. All of these are simple calculations.

The calculations for each point consist of convolving a mask with a horizontal line of pixels (1 high by n wide). The particular mask used is not specified by the global edge detection process itself, and should be chosen to

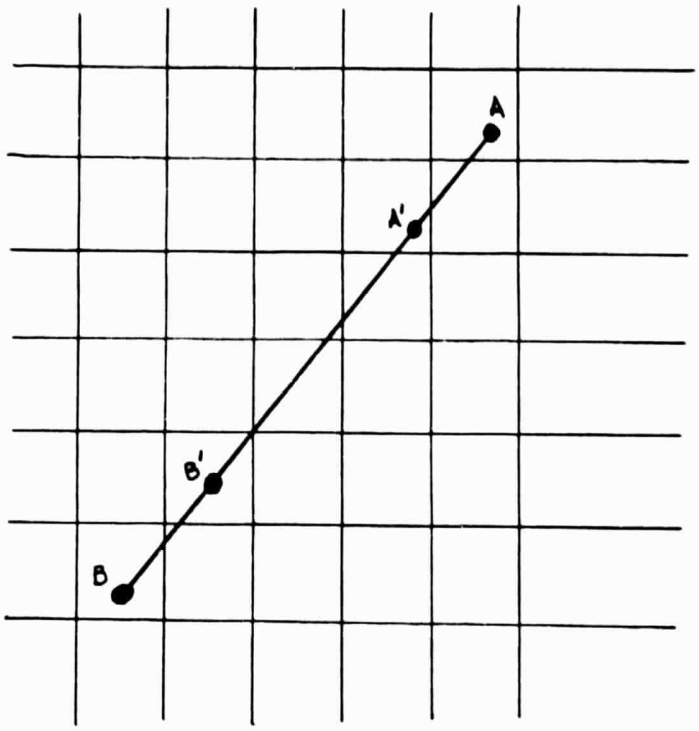


Fig. 5. The calculations for the line A'B' are a subset of those for the line AB, so that a great saving in computation time is made by avoiding duplicate calculations.

give optimum detection, consistent with efficiency, for the particular problem at hand. In the simplest version of the boundary initialization routine, this mask is just a simple difference operator.

The result of this convolution operation is used in different ways depending on the circumstances. If a line has been projected to pass through the point, then the convolution value is accumulated on the line, less a minimum value for the difference. If the accumulated value since the most recent high accumulation becomes sufficiently negative, then the line is ended back at the most recent high point.

Similarly, if there is no line projected through the point, then the convolution output is added to various local sums and the result is compared to a threshold for new line starting. The local sums are basically the best tentative lines passing through the point, each associated with its best starting point. Thus a line is started when the accumulated convolution output (squared) is raised above a threshold; the line is started at the statistically "best" point above the line being scanned when the start was made.

This process is done for an x-direction scan and a y-direction scan, in order to detect all lines. The detailed operations made are all listed in Table la and Table lb.

We thus see that <u>global edge detection</u> is a process which looks for a whole line, rather than the local crossing of a line with a pixel. It is conceptually more accurate, and, when the problems of associating local pixels are considered, it is easier to use and program than is local edge detection. It is this process that is used in the boundary graph initialization routine.

Adjustment Routine

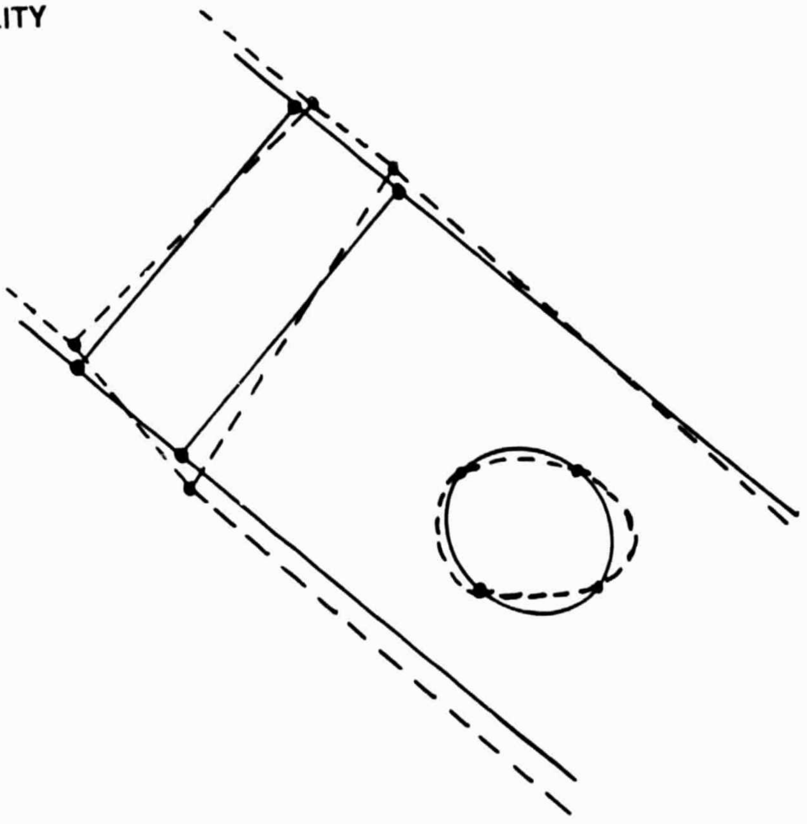
The adjustment routine corrects the boundary graph generated by the initialization routine. The initialization routine will give approximate, first-guess positions to the lines in the image, estimated at $\pm$ 1 pixel error. The adjustment routine will make small-scale adjustments to these initial boundary positions to achieve a best-fit boundary graph. The adjustment routine calculates a set of shifts of the boundaries (or endpoints) using the exact mixing of the gray pixels on the boundary to determine the optimum boundary position. These corrections may be applied iteratively to give as accurate an adjusted graph as desired.

Suppose we are given a boundary graph. We can calculate the expected luminance of any mixed pixel (a pixel straddling the boundary) from the coordinates of the boundary. We may then calculate the deviation of the actual brightness from its expected value. Summing some measure of this deviation along the line gives a measure of the fit of the line to the data. Adjustment of the line can then proceed using derivatives of this measure of fit.

The simplest measure of fit is least-squares. This measure can be posited directly, or derived from a maximum-likelihood approach. The latter is preferred, since it also allows the handling of more general situations.

Figure 6 shows an example of the boundary adjustment problem.

The adjustments of each line are not independent, since



---- Original lines (from initialization)

——— Final, adjusted lines

Fig. 6a. This figure shows the effect of boundary adjustment on an image. The original lines and their adjusted positions are both shown.

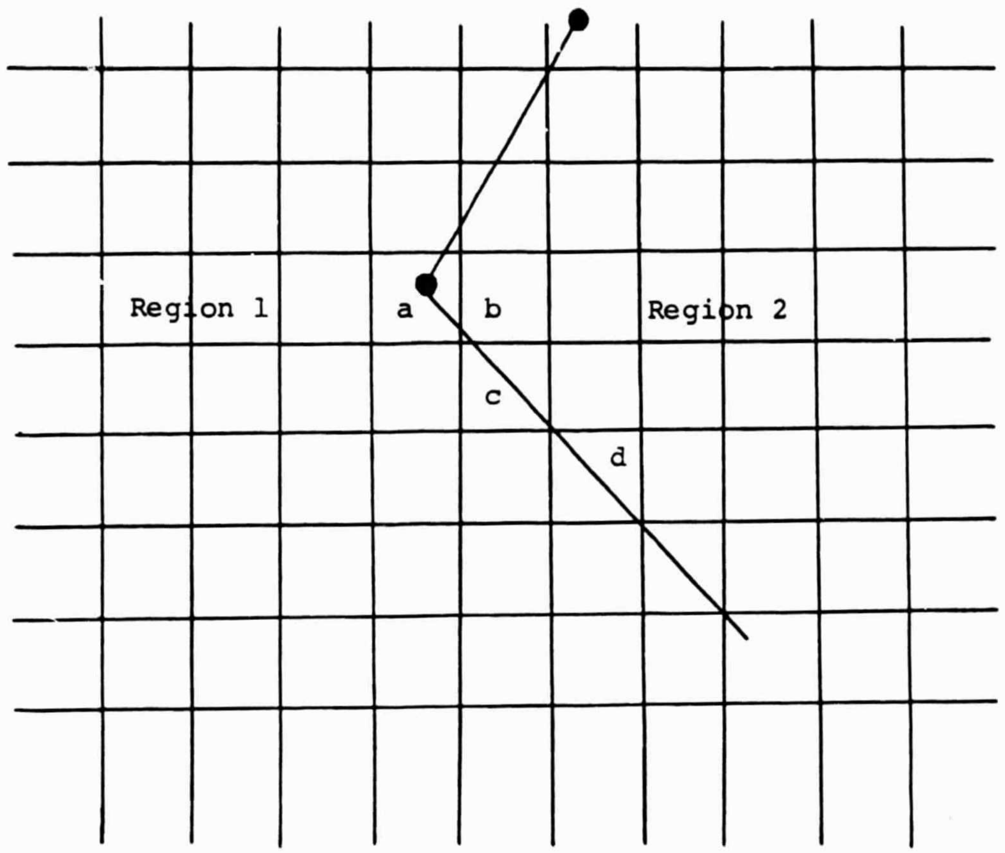


Fig. 6b. Each of the pixels on the boundary can be divided into a part belonging to Region 1 and a part belonging to Region 2. For example, pixel a is mostly Region 1, pixel b is mostly Region 2, pixel c has slightly more Region 1, and pixel d is evenly divided. These divisions can be calculated in terms of the parameters of the lines, and the expected greyness or color compared to the observed values, enabling an adjustment of the line position.

many lines will have common endpoints. Thus an inner iterative loop is required, given the boundary fit derivatives, to find the best (linear) adjustment of the boundary to the actual data. In an outer loop, we recalculate the fit and its derivatives and then recycle through the inner loop for solving the linear equations defined by the derivatives of the measure of fit.

The final result of the boundary adjustment program is a boundary graph with coordinates which give a best fit to the boundary position for the grey levels observed in the data. This is the closest approximation to the position which is available in the data on a statistically significant basis. Boundary adjustment needs to be done only where precise positioning information is required, and should typically give results accurate to $\pm$ .1 pixel.

Faint and Diffuse Line Detection

Another correction to the output of the boundary initialization routine is the location of faint and diffuse lines. The initialization routine will detect only fairly sharp, distinct lines. This results from its need to do only local processing in order to quickly scan an image. (A way of handling this problem, which will take care of most of the situations examined below, is to group the pixels into squares of 4, 16, 64, etc. and do a boundary analysis of these. This would find most of the faint and diffuse lines, but there would be difficulties in relating the various sizes=levels of abstraction. The procedure defined below will find these lines more completely.)

There are two types of boundary which produce insufficient gradient to trigger the edge detector in the initialization routine. These are faint boundaries, which do not have a large enough step to be detected in scanning across on a single line, and diffuse boundaries, where the step is spread across so many pixels that it is locally undetectable. It is useful to have a separate process for detecting these types of edges, augmenting the boundary initialization routine.

The procedure used is related to, and as a program, uses, the procedure below for detecting continuous variations. Each region is divided into two sub-regions, and and each of these into two further sub-regions, etc. down to the pixel level. If the two parts of a region are significantly different in brightness, then the possibility exists that the region is divided by a boundary through it. It is also possible, however, that the region just has a continuous variation in brightness across it. The program must then locate the best position for an edge and determine if that edge is a better hypotheses than the assumption of a gradient.

The best position for the boundary may be determined by trying it in various positions and determining which fit is best. For local changes in positioning, we may simply look at the pixels adjacent to the boundary and determine if the boundary would be better placed on the other side of them. This same process may be applied to the larger groups of pixels in the tree generated by the continuous variation system, giving more global aprroximations to the boundary position.

We may, therefore, find the best boundary position by the following process: first, assume that the boundary lies between the two subregions which were found to have a significant difference in brightness. Look at the highest-level subdivisions of these regions. Find any of these subdivisions which is next to the boundary and is spectrally more like the region on the other side of the boundary. Move the boundary to put the region on the other side. Repeat this process using smaller subdivisions until the pixel level is reached.

The resulting boundary position, achieved by successive motions of the boundary, is a simple best fit to the optimum position, and the degree of fit achieved by it can be compared to the degree of fit given by a gradient or other continuous variation fit. Of course, the boundary fit contains additional parameters, and so will naturally give a somewhat better fit. This must be compensated by _a priori_ biases in the Bayesian decision rule used to decide which alternative to choose, or by some related method.

If the boundary approach yields the best fit, then a new boundary is created and put into the boundary graph. The new regions created by this process are checked for further divisions. In this way all the faint and diffuse lines are discovered and placed in the boundary graph. This process, with proper setting of controls, should find all the statistically significant boundaries in the image.

A special question arises for diffuse lines, whether discovered by this method or by the boundary initialization routine. Any diffuse line will have a _form factor_, that is, a profile of brightness changes across the line. A "standard" form factor would be tanh(x) or some similar smooth step function. However, most diffuse lines will have a different brightness shape. In addition, the width and shape of the line vary along its length. This may be most clearly seen in a shadow from a corner, or an object rising out of the surface on which the shadow is cast. As the shadow gets further from the edge casting it, the width of the shadow will increase, from zero where the shadow is next to the shadow caster.

Thus some explicit account should be taken of the form factors of diffuse lines. This may range from a fixed, simple shape invariant along a line, to complex structures allowing changes of width and shape along the length of a line, and allowing complex line shapes. For most applications the former, simpler method is probably the best, but for some applications the more complex methods may apply.

In summary, one of the extensions of the boundary graph system, beyond the initialization, is the location of faint and diffuse lines, which are then included in the boundary graph. This is done by taking information generated by the continuous variation system, and determining whether a proposed continuous variation is in fact a boundary. An additional point is that diffuse edges must be modeled in some fashion, which might be quite complex.

Continuous Variation and the Zonal Method

Continuous variation is relatively smooth variation within a region. Since the boundaries define a region, there are no boundaries in the area to be analyzed by the continuous variation model.

This variation may give a constant value across a region, a gradient, or various kinds of local variation or mottling. We wish to divide the region into a minimum number of zones, across which the brightness is constant or linearly varying. This will allow description of the image brightness with a minimum number of parameters.

The zonal method divides a region into successively smaller zones by binary division in the longest dimension. This forms a tree of subdivisions of different sizes. Only part of this tree is used for the zonal process. If two subregions have essentially the same value of a parameter they are not retained as separate from the parent region. Of course the parent region itself may be combined with another parent region on the same level to form a super-parent region. The division as to whether to drop a level of detail is made on the basis of statistical significance of the difference between the two regions proposed to be combined. In a slightly more advanced version of the algorithm, tests may be made as to whether to combine zones with different parent zones, thus forming statistically uniform zones of variable shape.

When the zone analysis is complete, we are left with a tree of zones, some large, some small, where there is always a statistically significant difference between different remaining branches. Figure 2 shows an example of the zonal analysis on a region.

This method allows us to compact data maximally without throwing away any statistically significant information. The zonal information gives a representation of the continuous variation as it can be recognized under tests of statistical significance. The zonal analysis may be done using the x and y gradients in addition to the parameter being analyzed itself. This will produce a smoother model of the variation of the parameter mean.

The continuous variation procedure thus models the data with a minimum set of independent values. A value or gradient is assigned to each zone of varying size, and these make up the model of the variation.

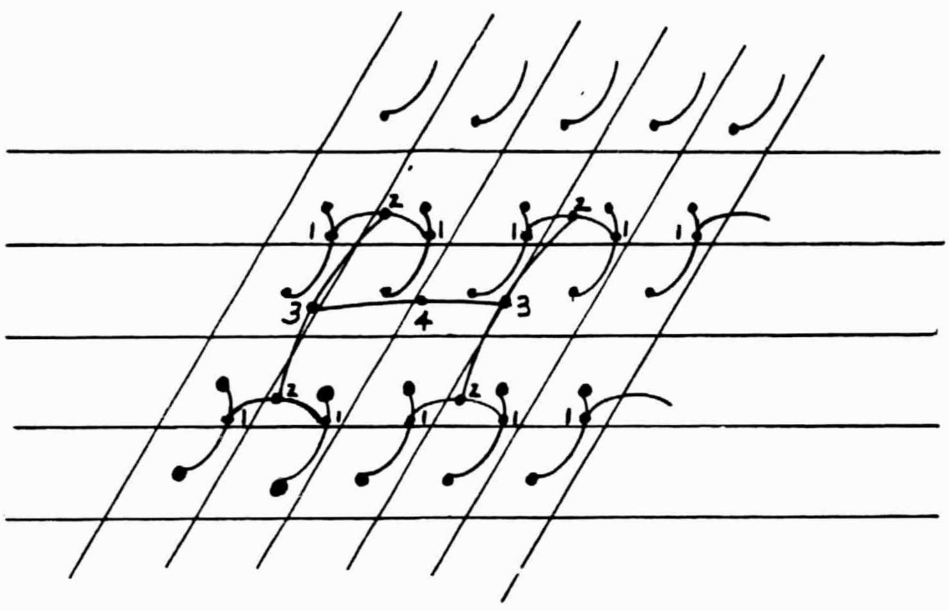


Fig. 2. Zonal accumulation scheme. Each pixel is averaged into a first-level (two pixel) group denoted by 1. These are accumulated into (four pixel) level 2 groups denoted by 2, and so on up to the highest level in the picture, denoted by 4 and containing 16 pixels. Tests are made at each level to determine if the pixels belong together. The accumulation process must be done in a slightly different way when the area has a more complex, limited shape.

Texture Detectors

Besides the detectors for the boundary analysis system and the continuous variation system, which are described elsewhere, there are special detectors for texture components, and higher-level detectors for periodicities, repeated figures, special angles, etc. These detectors are used to gather additional or higher-level information about the image.

The textural detectors gather information about a specific region; information from multiple regions may be integrated together if there are elements of common texture. Basic texture analysis consists of evaluating various local n-point operators on the region. A typical 2-point function, the power spectral density, gives the correlation of points at various relative positions (distances, angle) to each other. This information may be further reduced by looking at the isotropic (angularly uniform) falloff of the power spectrum: does it fall off as a power law? with what exponent? are there subsidiary peaks? at what distances? etc. The isotropic (all-directional) information in the power spectrum may thus be reduced to a few, or many parameters, depending on the circumstances and general usage of the vision system.

The non-isotropic information in the power spectrum may be analyzed in a similar fashion. If the power spectrum falls off differently in different directions, then this information may be reduced to vector information about the power spectral components (that is, to x and y components of the power spectral parameters.) For example, in a region of one-directional fibers, the correlation will have a (cos 2θ, sin 2θ) type dependence, with high correlation along the direction of the fibers, and low correlation, with subsidiary peaks, in the cross direction. The regularity of the fibers is shown by the strength and sharpness of these subsidiary peaks. In a normal fabric the power spectrum autocorrelation will have 4 peaks as we go around the circle of directions. That is, it will be modelled by a (cos 4θ, sin 4θ) type of angular dependence. Looking at the various parts of the non-isotropic variation and parametrizing them will give measures of the textural information. As in the case of the isotropic information, the number and type of the parameters used is dependent on the type of scene being analyzed.

The simplest power spectral components may be analyzed using only the eight pixels adjacent to any level. The accumulation of these correlations for all the pixels in the region will give a four-parameter description of the texture. (There are only four parameters since the auto-correlation function is even, and averaged correlations on opposite sides of a pixel will be the same.) In most cases, however, the scale of the textural features will be larger than a pixel, so that power spectral information must be extracted at larger distances and lower spatial frequencies.

We may use two-point functions other than the power spectrum to analyze the texture of a region. The power spectrum is derived from the autocorrelation function and is basically bilinear. Other measures, such as looking at extremal values, at mathematical transformations of the variables, or even at bihistograms, may be used in deriving textural parameters. Again, the particular reduction of the data used depends on the application.

Of course, relations may be made between more than two pixels. However, the number of parameters or dimensions of parameters grows astronomically as the number of points used increases, and severe reductions must be made in the parameter space. For certain applications special n-point functions may be devised, if enough is known that certain n-point configurations are expected. Otherwise abbreviated forms must be used.

The simplest n-point functions are the four-point quadratic forms of localized two-point functions. For these, we take a local two-point, generally non-isotropic, function and sum it over areas of intermediate size. We then square the result, and add the squares over the entire region. An example of this is the textural analysis of a region with fiber patches of varying direction, for example, a spruce tree. The fibers do not have any common direction, and would not give any large non-isotropic parameter. However, in a local sense the region is certainly composed of fibers, and the four-point function described will measure this.

Other types of n-point detectors are based on edge detection or other "macroscopic" types of analysis. Complete analysis of a fiber area falls in this class. Since the edge detection has generally been done already, these measures may be computationally efficient.

The variety of textural information that may be gathered is immense. A few basic measures may be selected as having major importance, but many others may appear in special cases. Ultimately, a texture system will probably take into acccunt the particular texture peaks, valleys, and other features appearing in a given region on an adaptive basis. The problem of texture analysis is basic and incompletely understood.

Higher Level Detectors

Higher level detectors are detectors which use the output of one of the original or low-level detectors to make additional analyses. Usually the lower-level detector used is the boundary detector.

Numerous analyses can be made based on the boundary graph. There may be some preferred orientation of image lines (vertical, North, etc.) There may be special angles of incidence in the picture, such as a tendency for lines to meet at right angles. The most important of these is the location of parallel lines. Parallelism is an invariant no matter what angle one views the picture from. Parallel lines occur in almost every application, from Landsat data and optical character recognition to locating parts in a bin (for example, the edge of a punched part.) Parallelism is also one feature detected in the early stages of the human visual system (reference not available). Parallelism is thus the primary higher-level detector. Derived from parallelism is the location of periodic or repeating structures.

There are a number of more complex detectors based on the boundary graph. Some of these additional detectors and processing include: the detection of occluded or partially covered up lines, locating multiple copies of a figure in a scene, analysis of three-dimensional rotations, matching figures to templates, analysis of shadows, analysis of general visual scenes, etc. These procedures are outside the provenance of this report.

Use of the Scene Model in Registration

The scene model described in this report was originally designed to aid in the registration of two images to subpixel accuracy. In this section we show how this may be done.

When the boundary initialization and adjustment routines are both used, the result is a very precise set of boundaries. The subpixel accuracy of these boundaries is the basis of the subpixel accuracy of the registration.

Registration based on the boundary graph is somewhat different from that based on the point-by-point correlation. For boundary graph registration, corresponding points on the two images are identified using context and approximate position. When a sufficient number of these have been identified, we can use them as bases to match up all corresponding lines in the two graphs. We then have a qualitative alignment of the two graphs.

The qualitative alignment can be used to generate a registration displacement by calculating the displacement of various points and averaging over regions of some size. This displacement can be used to produce the proper resampled image.

Boundary graph registration is better than correlation registration in several ways. Primarily, the identification of features from one image to the other is positive, and is not simply "there is an edge here." Features may be identified in the second image far from their positions in the first image. Since each feature is separately correlated, we may closely map the cross-correlation surface, and appropriately deweight features which have moved excessively. In general, the correlation process is under much better control, and greater selectivity of correlation is possible.

Other advantages of the boundary graph approach include: 1) It is possible to eliminate boundaries appearing on only one graph from consideration. 2) Some lines may be eliminated or deweighted directly. 3) The uncertainty in line position may be used quantitatively. 4) Historical data may be accumulated for an area. 5) Registration may be made to a properly input map.

The boundary graph may also be used to perform boundary oriented resampling. In this case, since the boundary positions are known, the form of resampling used to produce a registered image may be modified near a boundary to give an optimum fit to the relocated boundary. This system would not smear boundaries during resampling, but would produce sharp boundaries in the resampled image. Such an image will be closer to what one would expect had the original image been made on the resampled grid.

Summary

This report has outlined a general-purpose first-stage visual system. This system was originally devised for the processing of Landsat data for the NASA-NOAA-USDA AgRI-STARS Project, which attempts to gain agricultural information. However, after the system was fully conceptualized, comparison with other visual problems in industry and in other areas revealed that this basic process applied there as well. This presentation has been oriented towards this general usage of the system.

There are three major parts to this recognition system: the boundary analysis subsystem, the continuous variation analysis subsystem, and the miscellaneous features analysis, including texture, line parallelism, etc. These subsystems all work together. For instance, part of the boundary subsystem uses the output of the continuous variation subsystem. Similarly, the parallel recognition subsystem uses the boundary output; and the texture output will be further analyzed in both the boundary and continuous sections.

The boundary analysis subsystem is the most important. This system detects all the boundaries and lines in the image, and builds a boundary graph. This divides the image into regions, corresponding to fields in Landsat data and surfaces of an object in analyzing an every day environment.

Complete boundary analysis is done by using three routines, the initialization routine, the boundary adjustment routine, and the faint and diffuse line detector. Of these, the initialization routine is used to get the initial boundary graph. The other two are applied, possibly recursively, to the initial output, to correct it to make an optimum fit. These correcting routines do not need to be applied except in cases where the improved output is required. Most situations are analyzed perfectly well by the initialization routine, and will not require any corrections. The faint and diffuse line detector relies on information from the continuous variation system.

The continuous variation system finds gradual variations which are not well approximated by a boundary structure. These include gradients, mottling and similar features. The continuous variation is analyzed by successive binary division of the image, looking for significant differences. This allows the modelling of local, strong variations, as well as larger, more gradual ones.

Texture is an important part of many analyses. The actual measures of texture employed will vary from problem to problem, but in each case they will result in local measures of texture. These may be processed further by the boundary and continuous variation subsystems to detect boundaries and gradual variations in texture.

Additional analyses may be made on the output of the boundary routines, looking for parallel lines, and other spcial angles. When taken to an extreme, this becomes shape analysis, which is beyond the scope of this program, although a true shape analysis would use the output of this program as input.

The analyses done by these systems give a good representation of the image. The model of the scene resulting from these analyses will give brightness values not statistically discernible from the original data, except for the fact that they are much less noisy. Thus the model closely describes the data. It is also in a form which makes it much more possible to answer questions about the scene.

This model, especially the boundaries, is especially useful in the registration or rectification of images, since the boundaries may be used directly in the cross-correlation of two images. This method is under much better control than is the ordinary method of cross-correlation. It also can produce a much higher accuracy. Correlation with maps (rectification) is easy in this framework. Thus the accurate boundary model produced by these routines is the basis for a highly improved correlation routine.

By creating an accurate model of the scene, we make possible detailed analysis of the scene and its features. For Landsat data, we have a separation of the scene into fields. For ordinary objects, we have the edges and outlines by which they can be recognized and located. Thus the calculations done here constitute a basic first step in the analysis of an image.

Bibliography

Castleman, Kenneth F., Digital Image Processing, (Prentice-Hall, Englewood Cliffs, N.J., 1979)

Duda, Richard O. and Peter E. Hart, Pattern Classification and Scene Analysis (Wiley, New York, 1973)

Pratt, William K., Digital Image Processing, (Wiley, New York, 1978)